NASA/TM—2012-217272

Methods for Computationally Efficient Structured CFD Simulations of Complex Turbomachinery Flows

Gregory P. Herrick
Glenn Research Center, Cleveland, Ohio

Jen-Ping Chen
The Ohio State University, Columbus, Ohio

April 2012

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to *help@sti.nasa.gov*

- Fax your question to the NASA STI Help Desk at 443–757–5803

- Telephone the NASA STI Help Desk at 443–757–5802

- Write to:
  NASA Center for AeroSpace Information (CASI)
  7115 Standard Drive
  Hanover, MD 21076–1320

NASA/TM—2012-217272

Methods for Computationally Efficient Structured CFD Simulations of Complex Turbomachinery Flows

Gregory P. Herrick
Glenn Research Center, Cleveland, Ohio

Jen-Ping Chen
The Ohio State University, Columbus, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

April 2012

This work was sponsored by the Fundamental Aeronautics Program at the NASA Glenn Research Center.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076–1320

National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Available electronically at http://www.sti.nasa.gov

Methods for Computationally Efficient Structured CFD Simulations of Complex Turbomachinery Flows

Gregory P. Herrick
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Jen-Ping Chen
The Ohio State University
Columbus, Ohio 43210

Abstract

This research presents more efficient computational methods by which to perform multi-block structured Computational Fluid Dynamics (CFD) simulations of turbomachinery, thus facilitating higher-fidelity solutions of complicated geometries and their associated flows. This computational framework offers flexibility in allocating resources to balance process count and wall-clock computation time, while facilitating research interests of simulating axial compressor stall inception with more complete gridding of the flow passages and rotor tip clearance regions than is typically practiced with structured codes. The paradigm presented herein facilitates CFD simulation of previously impractical geometries and flows. These methods are validated and demonstrate improved computational efficiency when applied to complicated geometries and flows.

Introduction

With advances in computational science and technology, simulations of more complex turbomachinery geometries and flows become more feasible and appropriate. The application of CFD to the study of compressor stability has introduced new challenges of infrastructure and resource availability; these challenges have motivated the research discussed herein.

The research which follows documents the enhancements to a state-of-the-art turbomachinery computational fluid dynamics code to facilitate further research into compressor stall inception and other complex turbomachinery flows. Modifications are implemented to increase the speed and computational efficiency of the code while retaining accuracy and backward-compatibility with existing input/output data. An analytic viscous flux Jacobian computation is installed, replacing the previous numerical computation of the viscous flux Jacobian. A “multiple-blocks-per-computational-process” data structure framework is implemented to facilitate efficient load balancing for large-scale parallel computations; this enhancement is available at the user’s discretion and control, and the user may forgo use of this feature with no loss in other functionality (but with additional computational efficiency unrealized).

This “multiple-block” scheme, as “multiple-blocks-per-computational-process” shall be condensed in reference hereafter, is motivated by the desire to achieve greater computational efficiency and load balancing for CFD simulations containing disparately sized grid blocks, because many modern turbomachinery geometries and their associated flows have small, local regions laden with complex fluid mechanic phenomena. These local regions can be of much lesser geometric/computational extent than the global flow channels, which historically has motivated the use of models rather than grids in the local regions.

Nomenclature

A^v	Viscous Flux Jacobian Matrix
A^{v+}	Positive Viscous Flux Jacobian Matrix
A^{v-}	Negative Viscous Flux Jacobian Matrix
e	Total energy per unit volume
F, G, H	Inviscid Flux Vectors (curvilinear space)
F^v, G^v, H^v	Viscous Flux Vectors (curvilinear space)
k	Generalized curvilinear coordinate
Q	Conservative variable vector (curvilinear space)
$\overline{Q}$	Conservative variable vector evaluated at cell interface
S	Body force vector due to rotating frame
u,v,w	Absolute velocity (Cartesian space)
x,y,z	Spatial Coordinate (Cartesian space)
Δ	Increment, change
ε	Very small number
Ω	Rotational speed
ρ	Density
τ	Temporal coordinate (curvilinear space)
ξ,η,ζ	Spatial coordinate (curvilinear space)

Flow Solver

TURBO is a physics-based simulation tool for multistage turbomachinery. The solver computes the fluid conservation laws—the compressible RANS equations of Equation (1)—without ad hoc modeling of any flow phenomena other than models required for turbulence.

$$\frac{\partial Q}{\partial \tau}+\frac{\partial\left(F-F^v\right)}{\partial \xi}+\frac{\partial\left(G-G^v\right)}{\partial \eta}+\frac{\partial\left(H-H^v\right)}{\partial \zeta}-S=0 \tag{1}$$

where

$$Q=J\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ e\end{bmatrix}, S=\begin{bmatrix}0\\ 0\\ -\rho w\Omega\\ \rho v\Omega\\ 0\end{bmatrix} \tag{2}$$

This code solves the unsteady Reynolds-averaged Navier-Stokes equations and a decoupled k-ε turbulence model developed by Zhu and Shih (Ref. 1). To facilitate rotor-stator interaction studies, TURBO employs a sliding interface technique implemented by Chen and Barter (Ref. 2) in which conservative variables are interpolated across blade row interfaces. The code is implemented in a portable, scalable form for distributed-memory parallel computers using MPI message passing. The parallel implementation employs domain decomposition and supports general multi-block grids with arbitrary grid-block connectivity. The solution algorithm is a Newton iterative implicit time-accurate

scheme with characteristics-based finite-volume spatial discretization. The Newton subiterations are solved using a concurrent block-Jacobi symmetric Gauss-Seidel (BJ-SGS) relaxation scheme. Because all of the fundamental fluid mechanics are computed, the code is capable of capturing the nonlinear characteristics of the flow fields of interest. With the actual modeling of the grid movement of the blade rows in relative motion, this code is capable of computing the unsteady interactions between blade rows. Details of the flow solver are given by Chen and Whitfield (Ref. 3). The original approach to parallelization for large-scale, complex problems is discussed by Chen and Briley (Ref. 4); modifications and enhancements to this parallelization scheme are described forthwith.

Enhancements to Flow Solver

To facilitate computationally efficient solutions of large-scale and increasingly complex turbomachinery flow conditions, effective algorithms must be developed and implemented. These algorithms must have good performance, high-efficiency CPU utilization, efficient memory utilization, and exhibit portability and scalability.

Analytic Viscous Flux Jacobians

Good (improved) algorithmic performance is addressed in this study through the implementation of analytic viscous flux Jacobians. To facilitate solution of the compressible RANS equations through numerical methods of linear algebra, the fluxes of Equation (1)—nonlinear functions of time and space—must be linearized. Using methods of Briley and McDonald (Ref. 5), linearization is achieved approximately via Taylor series expansion, as shown in Equation (3).

$$K^n \approx K^{n-1} + A\left(Q^{n-1}\right)\Delta Q^{n-1} \tag{3}$$

where $A(Q)=\partial K(Q)/\partial Q$ is the flux Jacobian, and the superscripts here indicate time step indices. Since its inception, TURBO has utilized analytic definitions for the inviscid flux Jacobian terms, but numerical definitions for the viscous flux Jacobian terms (Ref. 6):

$$A^v(Q) = \frac{\partial K^v(Q)}{\partial Q} \approx \frac{K^v(Q+\varepsilon\Delta Q) - K^v(Q)}{\varepsilon\Delta Q} \tag{4}$$

The numerical calculation of viscous flux Jacobians, though simple in derivation and implementation, proves to be quite computationally expensive, accounting for up to 20 to 25 percent of total CPU time in many simulations. Previously at Mississippi State University, Cox (Ref. 7) implemented a more efficient analytic formulation for viscous flux Jacobian computation; Cinnella (Ref. 8) had implemented these formulations in the General Aerodynamic Simulation Program in his (Cinnella's) prior work.

Using the thin-layer approximation to derive the components of the viscous flux vector (detailed in Refs. 7, 8, and 9), the terms are easily differentiated with respect to the Q vector, yielding the analytic viscous flux Jacobian. To facilitate common code structure and efficient performance of the analytic viscous flux Jacobian computations, these analytically-computed diffusive vectors are (arbitrarily) implemented in the manner of the Steger-Warming flux vector splitting concept employed with the convective inviscid flux Jacobians:

$$A^v = A^{v+} + A^{v-} \tag{5}$$

Enforcing the above condition into the temporal evolution linearization of Equation (3) yields

$$\left[\frac{\partial K(Q)}{\partial Q}\Delta\overline{Q}\right]_l = A_l^{v+}Q_l + A_{l+1}^{v-}Q_{l+1} \tag{6}$$

where barred values are evaluated at volume interfaces and non-barred values are evaluated at volume centers. For cell interface l between volumes l and l+1, positively propagating waves originate at volume l while negatively propagating waves originate from volume l+1. All properties employed explicitly at cell interfaces are determined by averaging property values from the two adjacent cells:

$$\overline{Q}_l = \frac{Q_{l+1} + Q_l}{2} \tag{7}$$

All property gradients at cell interfaces are determined using second-order central differencing of the property values from the two adjacent cells. Observing that $\Delta k = 1$ yields:

$$\left(\frac{\partial Q}{\partial k}\right)_l = Q_{l+1} - Q_l \tag{8}$$

The final expressions for the analytic viscous flux Jacobian may be found in Herrick (Ref. 9).

Efficient CPU Usage: Multiple-Block Scheme

Two critical characteristics of efficient parallel CPU utilization are the minimization of message-passing and the balancing of computational load. In order to minimize delays associated with message-passing, it is desirable to maximize serial operations within the solution algorithm of the grand computational domain and to attain equivalently sized messages among all simultaneous parallel communications. It is also desirable to achieve a computational load balance whereby all processes are working (or communicating) simultaneously. These quests are difficult to satisfy due to disparate sizes and shapes among geometries, varied boundary conditions, and differences in the flow field.

The enhancement of the parallelization of TURBO from its original single-block structure to the modified multiple-block structure is depicted in Figure 1. The original parallelization strategy for TURBO greatly enhanced serial-TURBO's flexibility and applicability to larger and more complicated geometries. With the single block per process infrastructure, a complicated geometry with many subdomains conveys a demand for a very large parallel computer cluster; with disparately sized blocks, the computational efficiency of the simulation suffers. A multiple-blocks scheme allows small blocks to coexist with large rectangular blocks on a single process without compromising computational efficiency.

To create the multiple-block infrastructure, the one-block/one-process equivalence of the original parallelization is dissolved. Process identifiers within the code, as with the original code, remain uniquely `MPI_COMM_RANK` within `MPI_COMM_WORLD`. However, the data structures within the program must incorporate the new premise of multiple blocks coexisting, each with a global identifier unique to the grand computational domain and a local identifier unique to the specific process on which it resides. In the original parallel code, the process identifiers (`MPI_COMM_RANK` within `MPI_COMM_WORLD`) also serve to uniquely dereference CFD data for a given block.

The modification to the parallelization scheme, seemingly, requires the implementation of an outer loop cycling through all the local blocks residing on each respective process. Though simple in concept, this is not strictly possible due the presence of numerous conditional statements (tailoring program execution for varied boundary conditions, block connectivities, flow regimes, etc.) which may unavoidably invoke communication with other blocks residing on other processes. Note that the user has

some control of the frequency of communication between blocks, although each round of communication may enhance convergence at the expense of communication time, with parallel communication more taxing than serial communication.

Whereas initially all communication between blocks was between processes, the new scheme allows for the communication partner to coexist on the given process. Because blocks are likely to be distributed to processes in different quantities, MPI collective commands (e.g., `MPI_REDUCE, MPI_ALLREDUCE, MPI_SUM, MPI_SCAN`) must be called only so limitedly to satisfy the most restrictive block-process distributions. Ergo, all collective communications must occur only once per routine, per process. However, because each block has its own boundary conditions, connectivities, and associated flow, all blocks must execute the commonly shared solution routines (within the local block loop) before the process, and its resident blocks as a group, may engage in parallel communication. Subtle re-ordering of command sequences has resulted in insignificant changes in numerical computation outputs.

Efficient Memory Utilization: Multiple-Block Scheme

Nearly *every* data structure in the code requires modification to enable coexistence of multiple blocks on a single computational process. Originally, fundamental CFD values like grid data (`x,y,z`) and flow data (`q,p`) are stored in explicit-shape arrays dimensioned using the ξ, η, ζ extents of a single grid block: This original memory management scheme is ideal for its intent with optimal memory consumption, minimal pointer indirection, and dearth of unnecessary computations to fully dereference CFD data.

When seeking to place multiple blocks on a process, efficient memory utilization is a critical concern. Though this scheme performs best with regard to computational efficiency (i.e., with minimal pointer indirection and minimal high-level computations in the dereferencing procedure) by adding a trailing subscript, each dimension of the data structure is sized to accommodate the maximum value of that respective dimension among the subsets (grid blocks) of data. Thus, much memory is wasted if small blocks should cohabit with large blocks on a given process. Explicit-shape arrays are also inefficient with memory management in instances where the different blocks on the process have different dimensions (spatial coordinate directions) of maximum extent.

Alternatively, ribbon vectors are one-dimensional arrays sized exactly to the requirements of their application; they are *perfectly* efficient with memory. However, the pointer arithmetic necessary to navigate the array's "implicit dimensions," performed by the compiler at a lower level with explicit-shape arrays, must be performed in the software code at a higher level, at a great expense of added execution time.

Derived data types offer the computational efficiency of lower level pointer arithmetic performed by the compiler like explicit-shape arrays and most of the memory-management efficiency of ribbon vectors, as each parameter (component) of the derived data type may be sized based on the dimensions of the relevant grid block. Using derived data types, the q-vector can be transformed from `q(1:5,1:ni,1:nj,1:nk)` to `q(nlb)%v(1:5, 1:ni(nlb), 1:nj(nlb), 1:nk(nlb))`. However, this scheme necessitates three levels of pointer indirection to dereference CFD data.

While derived data types allow for precise sizing of component arrays to suit the storage requirements necessitated by large grid blocks and small grid blocks cohabiting, and also allow for lower-level compiler-performed pointer arithmetic, they are neither as memory-efficient as ribbon vectors nor as computationally efficient with compiler pointer arithmetic (pointer indirection) as explicit-shape arrays. Additional memory is used when defining the derived data type: The compiler must construct and follow a map to proceed from the primary variable (subscripted by local block index) down to the component (the relevant CFD parameter) down to the specific grid location within that component. This mapping concept associated with derived data types consumes additional execution time due to two more levels of pointer indirection, as well as additional memory: Beyond the raw data, derived data types consume $24+12*N_{\text{dim}}$ additional bytes for each of the N_{dim}-dimensional components of the derived data type variable (for `REAL*8` quantities). The additional pointer arithmetic necessary to dereference the desired

quantity—though performed at the lower level by the compiler—reduces computational efficiency somewhat, as shall be documented in results.

Table 1 summarizes performance parameters for the three data structure philosophies discussed herein.

Portability and Scalability

Throughout this study the code has performed efficiently on several machines at various supercomputing centers across the United States. TURBO, in its original and modified states (described in Table 2), has performed well on jobs consuming 4 to 234 processes in the course of this study. Version 3 represents the baseline, single-block code with numeric viscous flux Jacobian computations; V4 adds analytic viscous flux Jacobian computations, and V4+ introduces the multiple-block infrastructure.

Validation and Benchmarking

With all versions, TURBO distributions include a test suite to assist users in validating their TURBO installation and compilation. This test suite includes four cases well-suited for benchmarking here. General descriptions of these test cases are listed in Table 3. These four cases are benchmarked with and without application of the new arbitrary load balancing capability. Detailed performance data for each case including Inlet Physical Mass Flow, Exit Physical Mass Flow, Inlet Total Pressure, and Exit Static Pressure, are provided in Tables 4, 5, 6, and 7.

With the continued research on developing and applying TURBO for more complicated turbomachinery concerns, two other cases are appropriate for this benchmarking effort: the GE TEC56 HPT Nozzle 1 and a full-annulus grid of NASA stage 35 with gridded rotor tip clearance regions. Table 8 documents timing performance of the TEC56 Turbine Nozzle after 100 iterations, and Table 9 conveys timing performance for the full annulus NASA Stage 35 after 150 iterations. In addition to total wall clock time, timings of critical routines and total TURBO computation times for each process are also listed.

As documented here and in further detail in Reference 9, numerical accuracy is preserved, particularly between versions V4 and V4+. Numerical accuracy is also preserved when redistributing the block/ process layout with V4+, an option previously unavailable. Mass flow has stayed consistent within 0.22 percent, while pressure-performance has maintained accuracy within 0.11 percent. The numerical accuracy of the analytic viscous flux Jacobians is retained within about 3.5 percent, as documented in Reference 9. From V3 to V4, computation time of the viscous flux Jacobians has been reduced by 50 to 75 percent. Additional changes implemented in the evolution of TURBO from V3 to V4 yielded total computation time reductions of 10 to 30 percent.

In general, when executed in a single-block-per-process mode, the multiple-block (V4+) code suffers about a 7 percent speed penalty (ranging from 2 to 12 percent in the benchmark examples) versus the V4 code. This is attributed to the additional, more complicated dereferencing process of the newly-implemented derived data types, which require three times the pointer indirections as the explicit-shape arrays used in the V4 code. In the course of development, this disparity had been as large as 30 percent; deconstruction and reconstruction of some routines reduced the overhead of repetitive multiple-level pointer indirections, and the code's performance improved to the current levels cited.

As documented in Tables 8 and 9, when used optimally, the multiple-blocks approach of TURBO_P.V4+ provides superior efficiency for large-scale parallel computation. The TEC56 turbine nozzle uses nearly 40 percent less process-time on seven processes than when it utilizes one process alone for each of its seventeen blocks; the Stage 35 simulation uses nearly 25 percent less process-time on 234 processes than when it is executed with each of its 306 blocks individually occupying a process. The Stage 35 timing data affirms the speed penalty introduced by the extensive pointer indirection of derived

data types in computationally intensive routines such as the Symmetric Gauss-Seidel loop. The eddy viscosity computation is also adversely affected when disparately-sized blocks are run in a single-block-per-process mode with the V4+ code, but this penalty diminishes greatly when the total computational volume counts are more equally distributed among all processes with the multiple-block capability.

Analysis of Complex Turbomachinery Flows

The multiple-block capability is motivated by the desire to simulate complicated turbomachinery geometries and their associated flows. Among the complex turbomachinery flows of primary interest are those associated with stall inception in axial compressors. Much computational research on compressor stall inception has been performed with TURBO (Refs. 10, 11, and 12), but these earlier simulations employed simplified, periodic models of the flows across the rotor blade tips. This simplification was dictated by the previous single block per process infrastructure of TURBO. With the critical role of clearance flows in the stall inception process, it is of interest to capture the clearance flows in greater detail using separate grid blocks. A full annulus grid of Stage 35 with gridded rotor tip clearance flow (see clearance grid in Fig. 2) is simulated and studied here.

When investigating compression system instability through CFD a significant benefit of the gridded rotor tip clearance region, made feasible with the multiple-block per process infrastructure, is reaped in the post-processing stage. With fully contiguous gridding about the airfoil's leading edge, suction side, trailing edge, pressure side, and clearance regions, continuous particle traces become much more easily attainable and more precise than when certain grid segments like clearance regions are modeled. In Figure 3, particle traces are depicted demonstrating the breakdown of clean passage flow. In time, clearance flow sweeps forward, blockage develops and intensifies into the tightly-recirculating, forward-swept flow blockages characteristic of stall inception. Further discussion of this flow condition and its inherent fluid mechanic phenomena is available in Reference 9.

Conclusions

Two methods for improving the computational efficiency of the TURBO code were presented here: an analytic computation of viscous flux Jacobians, and a more versatile data infrastructure to handle multiple blocks on each computing process. TURBO_P.V4+ now computes viscous flux Jacobian terms in 50 to 75 percent less time than TURBO_P.V3. As a result of this work, the user can arbitrarily control the load balance distribution of blocks on processes. With good load balance distribution, the new code can compute large domains comprised of large and small subdomains in 20 to 40 percent less process-time. The user need not employ the new multiple-block capability; in such cases, accuracy will be retained, but speed will suffer about a 7 percent penalty due to the complexity of TURBO_P.V4+'s new data structures. The modified code is benchmarked versus the existing TURBO test suite, as well as with cases representative of contemporary turbomachinery research. A brief presentation of axial compressor stall inception demonstrates the utility and merit of this multiple block per process infrastructure in capturing complicated turbomachinery flows.

References

1. J. Zhu and T.H. Shih, "CMOTT Turbulence Module for NPARC," Tech. Rep. CR-204143, National Aeronautics and Space Administration, Aug. 1997.
2. J.P. Chen and J. Barter, "Comparison of Time-Accurate Calculations for the Unsteady Interaction in Turbomachinery Stage," AIAA-1998-3292, American Institute of Aeronautics and Astronautics, 1998.
3. J.-P. Chen, and D.L. Whitfield, "Navier-Stokes Calculations for the Unsteady Flowfield of Turbomachinery," AIAA-1993-0676, American Institute of Aeronautics and Astronautics, 1993.
4. J.-P. Chen and W.R. Briley, "A Parallel Flow Solver for Unsteady Multiple Blade Row Turbomachinery Simulations," Tech. Rep. GT2001-0348, *ASME TURBO Expo*, New Orleans, Louisiana, 2001.
5. H. McDonald and W.R. Briley, "Solution of the Compressible Three-dimensional Navier-Stokes Equations by an Implicit Technique," in *Lecture Notes on Physics*, Fourth International Conference on Numerical Methods in Fluid Dynamics (Boulder, Colorado), vol. 35, pp. 105-110, Springer-Verlag, 1974.
6. D.L. Whitfield and L.K. Taylor, "Discretized Newton-Relaxation of High-Resolution Flux-Difference Split Schemes," AIAA-1991-1539, American Institute of Aeronautics and Astronautics, June 1991.
7. C.F. Cox, "An Efficient Solver for Flows in Local Equilibrium," PhD Thesis, Mississippi State University, Mississippi State, Mississippi, December 1992.
8. P. Cinnella, "Flux-Split Algorithms for Flows With Non-Equilibrium Chemistry and Thermodynamics," PhD Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, December 1989.
9. G.P. Herrick, "Facilitating Higher-Fidelity Simulations of Axial Compressor Instability and Other Turbomachinery Flow Conditions," PhD Thesis, Mississippi State University, Mississippi State, Mississippi, May 2008.
10. J.-P. Chen, R.S. Webster, M.D. Hathaway, G.P. Herrick, and G.J. Skoch, "Numerical Simulation of Stall and Stall Control in Axial and Radial Compressors," AIAA-2006-0418, American Institute of Aeronautics and Astronautics, 2006.
11. J.-P. Chen, M.D. Hathaway, and G.P. Herrick, "Prestall Behavior of a Transonic Axial Compressor Stage via Time-Accurate Numerical Simulation," Journal of Turbomachinery, vol. 130, no. 4, October 2008, pp. 041014.01—041014.12.
12. M.D. Hathaway, J.-P. Chen, R.S. Webster, and G.P. Herrick, "Time-Accurate Unsteady Simulations of the Stall Inception Process in the Compression System of a U.S. Army Helicopter Gas Turbine Engine," *2004 DoD High Performance Computing Modernization Project Users Group Conference*, Williamsburg, Virginia, June 2004.

TABLE 1.—DATA STRUCTURES FOR REAL*8 MULTIPLE BLOCK STRUCTURED CFD DATA

Memory management scheme	Additional memory consumed	Levels of pointer indirection	Additional high-level computations
Trailing subscript	$8ni_{\max}nj_{\max}nk_{\max}mlb - 8\sum_{nlb=1}^{mlb} ni_{nlb}nj_{nlb}nk_{nlb}$	1	0
Ribbon vector	0	1	3
Derived data type	24+12* N_{dim}	3	0

TABLE 2.—DESCRIPTIONS OF CODE VERSIONS

Version	Viscous flux jacobian	Blocks per process
V3	Numeric	Single
V4	Analytic	Single
V4+	Analytic	Multiple

TABLE 3.—TURBO TEST SUITE

Case	Blade rows	Grid blocks	Grid points
Flat Plate	1	4	10 168
Rotor 67	1	4	44 950
Stage 37 TS	2	5	498 270
Stage 37 P	2	17	1 696 566

TABLE 4.—OUTPUT DATA: FLAT PLATE, 10000 ITERATIONS

Code	Procs	m_{in}(kg/s)	m_{ex}(kg/s)	PT_{in}(Pa)	PS_{ex}(Pa)
V3	4	52.39049	52.38945	110760.12446	98538.99859
V4	4	52.39076	52.38780	110759.74378	98537.99857
V4+	4	52.39076	52.38780	110759.74378	98537.99857
V4+	2	52.39076	52.38780	110759.74378	98537.99857
V4+	1	52.39076	52.38780	110759.74378	98537.99857

TABLE 5.—OUTPUT DATA: ROTOR 67, 1000 ITERATIONS

Code	Procs	m_{in}(kg/s)	m_{ex}(kg/s)	PT_{in}(Pa)	PS_{ex}(Pa)
V3	4	34.61931	34.22768	101254.86667	115091.16823
V4	4	34.59931	34.20919	101254.81966	115087.98710
V4+	4	34.59930	34.20919	101254.82005	115088.00740
V4+	2	34.59922	34.20958	101254.80964	115087.68202
V4+	1	34.60212	34.21248	101254.71517	115086.74200

TABLE 6.—OUTPUT DATA: STAGE 37 TIME-SHIFT, 168 ITERATIONS

Code	Procs	m_{in}(kg/s)	m_{ex}(kg/s)	PT_{in}(Pa)	PS_{ex}(Pa)
V3	5	17.42277	16.97482	101350.42278	100618.38075
V4	5	17.42011	16.95894	101350.51642	100584.15500
V4+	5	17.42075	16.95985	101350.88000	100589.98477
V4+	2	17.42075	16.95985	101350.88000	100589.98477

TABLE 7.—OUTPUT DATA: STAGE 37 PERIODIC, 168 ITERATIONS

Code	Procs	m_{in}(kg/s)	m_{ex}(kg/s)	PT_{in}(Pa)	PS_{ex}(Pa)
V3	17	17.72801	17.23441	101347.33946	101507.27163
V4	17	17.71025	17.20281	101351.90559	101410.57214
V4+	17	17.70894	17.19907	101351.53512	101398.52356
V4+	7	17.70894	17.19907	101351.53512	101398.52356

TABLE 8.—TIMING DATA FOR TEC56 TURBINE NOZZLE, 100 ITERATIONS

Code	Processes	Viscous flux jacobians, (cpu-s)	Wall clock, (s)	Proc-time, (cpu-s)
V3	17	207.25	585	5815
V4	17	37.06	550	5481
V4+	17	37.73	569	5674
V4+	7	51.28	874	3476

TABLE 9.—TIMING DATA FOR NASA STAGE 35 FULL ANNULUS WITH GRIDDED TIP CLEARANCES, 150 ITERATIONS

Code	Processes	Gauss-seidel, (cpu-s)	Eddy viscosity, (cpu-s)	Wall clock, (s)	Proc-time, (cpu-s)
V4	306	198737	44118	2571	783268
V4+	306	200502	45260	2548	777200
V4+	234	234485	27470	2627	611666

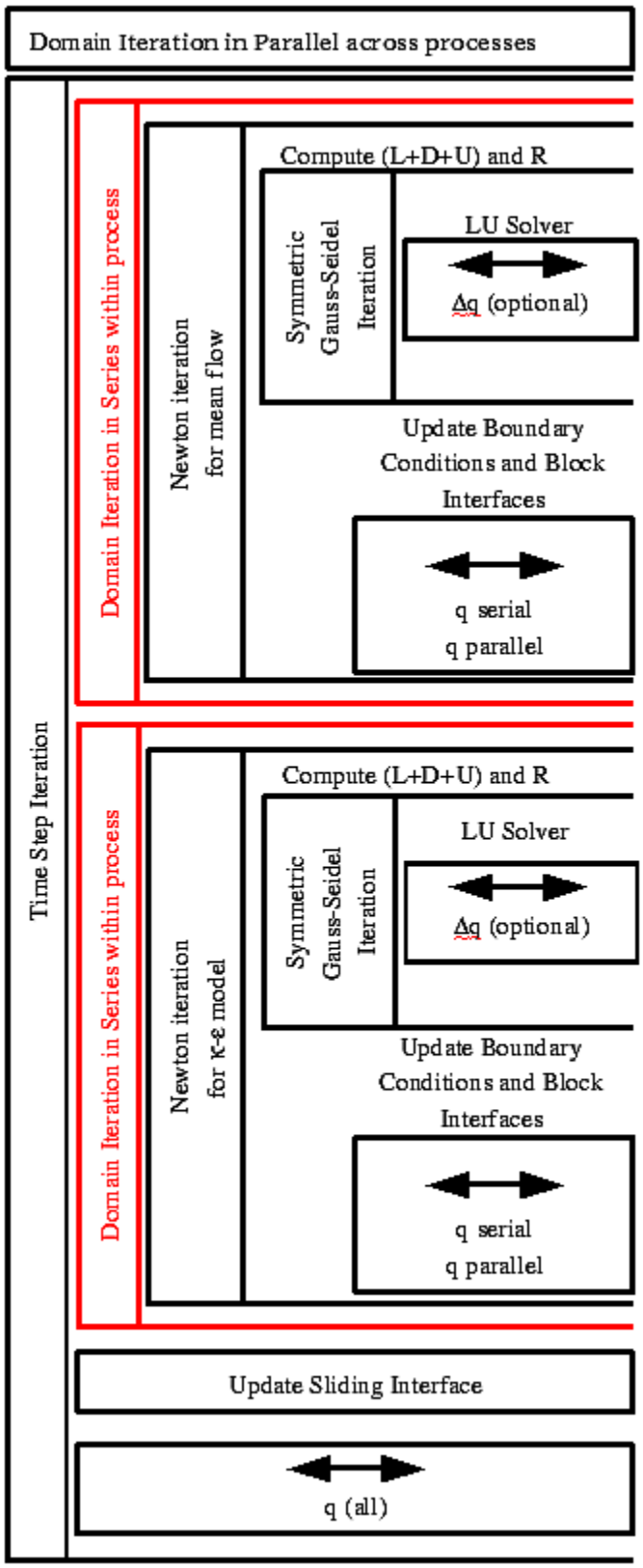

Figure 1.—Modification of parallel structure from single block per process (black) to multiple block per process (red).

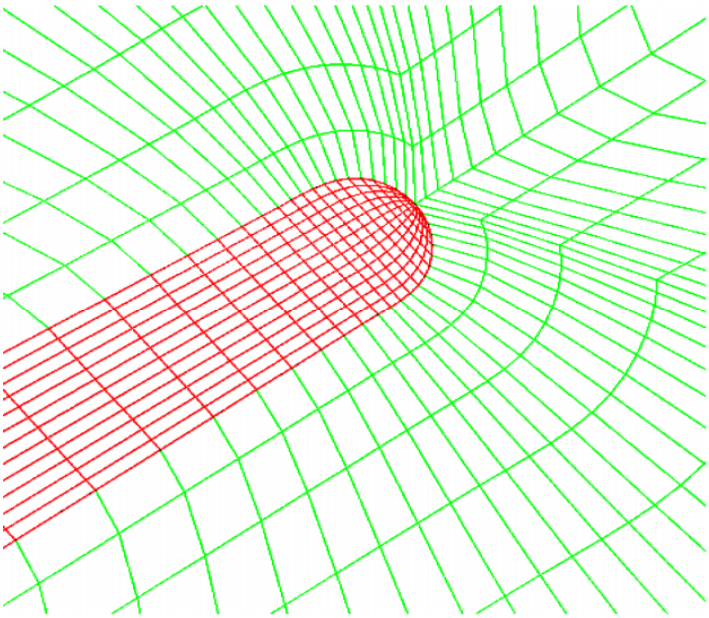

Figure 2.—Gridded rotor tip clearance region, shown plan-view.

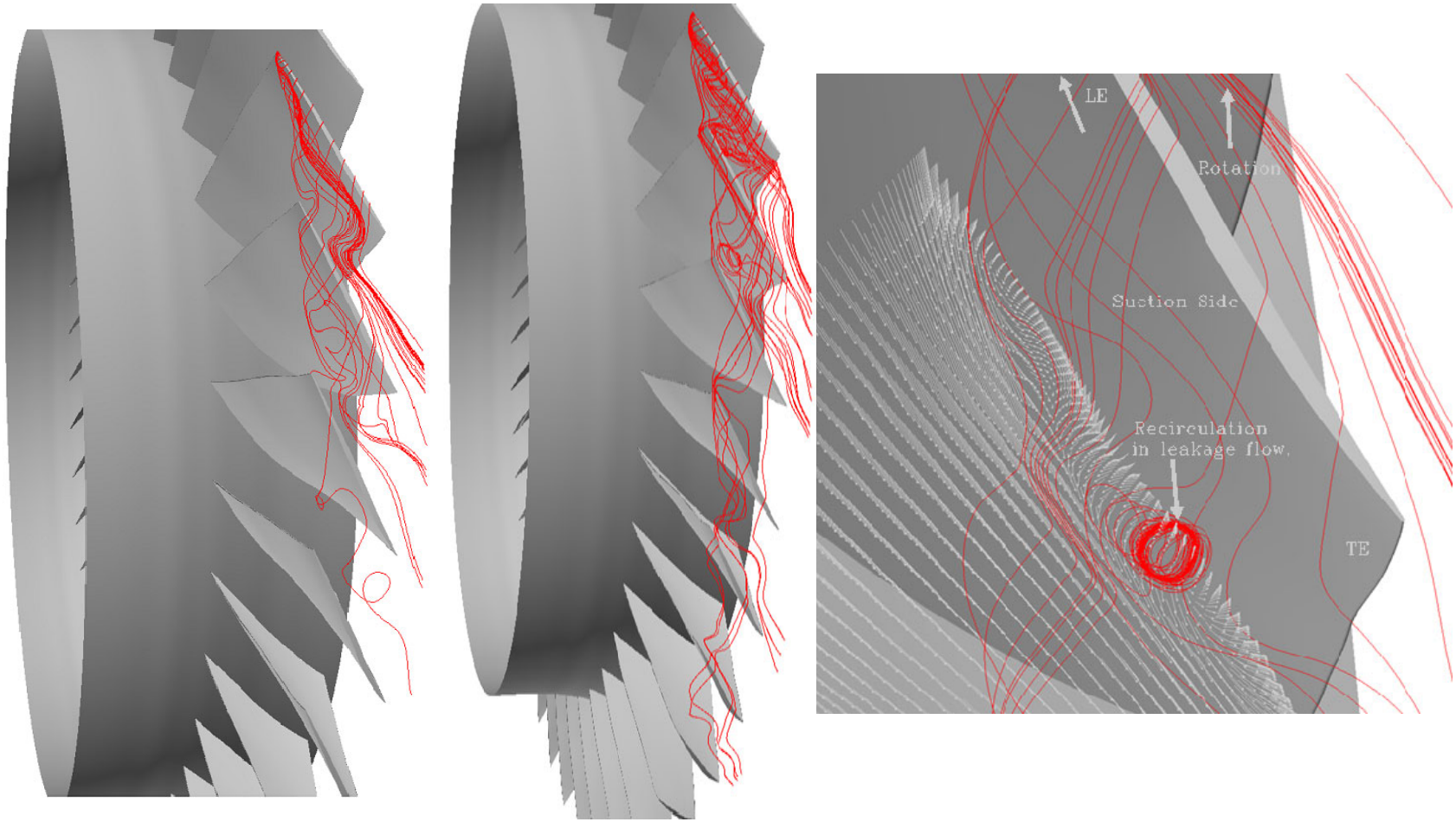

Figure 3.—The evolution of flow blockage and stall: (left) Minimal forward sweep of clearance flow, (middle) spreading and strengthening blockage, (right) intense recirculations and passage blockage.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-04-2012	Technical Memorandum	

4. TITLE AND SUBTITLE
Methods for Computationally Efficient Structured CFD Simulations of Complex Turbomachinery Flows

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Herrick, Gregory, P.; Chen, Jen-Ping

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
WBS 561581.02.08.03.21.03

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER
E-18028

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITOR'S ACRONYM(S)
NASA

11. SPONSORING/MONITORING REPORT NUMBER
NASA/TM-2012-217272

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Categories: 02, 07, 34, 61, and 64
Available electronically at http://www.sti.nasa.gov
This publication is available from the NASA Center for AeroSpace Information, 443-757-5802

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This research presents more efficient computational methods by which to perform multi-block structured Computational Fluid Dynamics (CFD) simulations of turbomachinery, thus facilitating higher-fidelity solutions of complicated geometries and their associated flows. This computational framework offers flexibility in allocating resources to balance process count and wall-clock computation time, while facilitating research interests of simulating axial compressor stall inception with more complete gridding of the flow passages and rotor tip clearance regions than is typically practiced with structured codes. The paradigm presented herein facilitates CFD simulation of previously impractical geometries and flows. These methods are validated and demonstrate improved computational efficiency when applied to complicated geometries and flows.

15. SUBJECT TERMS
Computational Fluid Dynamics (CFD); Geometry; Clearance; Efficient; Tip; Flow; Compressor; TURBO; Unsteady

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email:help@sti.nasa.gov)
U	U	U	UU	19	19b. TELEPHONE NUMBER *(include area code)* 443-757-5802